# THE TARGET

JULY/AUGUST 1980

-- an AIM65 newsletter

OLD BUSINESS

The first order of business is closing the books on Lunar Lander. The program description appeared in the May/June issue while the program itself was contained in the March/April 1980 issue. Hopefully this will tie all the pieces together and clear up any confusion regarding this program. I apologize for the gross error.

NEW BUSINESS

On page six of this issue is a letter from Jody Nelis which calls for more complete documentation on programs. The letter is presented as a brief guide for potential authors for any publication, but most of all, an AIM 65 user should discipline himself to follow the same rules for personal documentation as well. Returning to a program several months later will obscure why a particlar POKE was performed so the need of documenting everything is shown. I would like to emphasize that the letter should not be taken as a direct criticism of the authors mentioned. They are picked only as an example.

Rockwell has arranged a volume purchase of 6502 Software Design by Leo Scanlon. The price is only $7. 00 and includes postage and handling. You may order from Rockwell International, Microelectronics, RC 55 PO Box 3669, Anaheim, CA 92803. That's a good price! At the same address foreign AIM 65 users may be interested in the Rockwell newsletter, Interactive, it is available for $8.00 in US funds.

The Aim 65 has become quite intelligent these days. The AIM can now speak several languages including BASIC, XPLO, FOCAL, FORTH and PL/65.

Have you purchased a piece of software or hardware that you can not live without? Feel free to write and tell us why this is so. Spread the word.

It is apparent that a great deal of duplicate effort is being spent writing software. For each individual effort to solve a problem there has been at least ten or more duplicate efforts going concurrently. It's too bad that this newsletter couldn't be released at two week intervals. If it were possible then much of this duplicate effort could be eliminated. Any thoughts on this subject?

Until next time,

Donald Clem

Donald Clem




| | | |
|---|---|---|
| SOFTWARE | INTERFACING AIM BASIC | 2 |
| SOFTWARE | BASIC BANDAID | 4 |
| SOFTWARE | TAPE DIRECTORY | 8 |
| HARDWARE | USER VIA | 5 |
| HARDWARE | MEMORY | 5 |
| NEW PRODUCT | MACRO-ASSEMBLER | 10 |
| LETTERS | | 6 |
| TIPS | | 7 |

# Interfacing AIM BASIC

Jim Butterfield
14 Brooklyn Ave
Toronto Ontario
Canada M4M 2X5



AIM BASIC will go to a machine language subroutine by means of the USR function. The machine language subroutine will return to AIM BASIC with an RTS instruction.

The USR function is similiar to SQR,RND, etc. It can't start a line, but must be part of an expression. It takes an argument: USR(6) passes the argument, a value of 6, to the machine language program; and it returns a value: USR(6) might give back a value of 12 to BASIC. Both the argument and the returned value can be ignored, if desired. For example, X = USR(0) could be used to call machine language even though you don't care what value X becomes after the call, and even if the value of zero is ignored by the machine language program.

Single Routine

If you want to go to only one machine language routine, it's very simple. Just poke locations 4 and 5 with the address of the machine language (ML) routine, and USR will take you there. Example: you'll want to go to address 0F22 so in BASIC you say POKE 4,34:POKE 5,15. You need to do this once only and after that, any USR reference will take you to 0F22. (Note that hex 22 equals 34 decimal and hex 0F equals 15 decimal).

Multiple Routines

To get to various machine language subroutines, you could repeat the pokes to 4 and 5 before each USR reference. This is often convenient, but not lightning fast- BASIC POKEs are much slower than machine language.

If your routines come up in a certain order, you can have each ML subroutine set up the next. Since a POKE is roughly equivalent to a STA instruction, each routine could set up addresses 4 and 5 for the appropriate next USR entry address.

For example, assume that our previously set call to 0F22 is an initialization routine, and that all following calls should be to 0F4E.

We could code:

```
0F22 A9 4E  INIT  LDA #<ENTRY
0F24 85 04        STA $04
```

... and continue with our initialization. In this case, we don't need to change the high-order address in location 5, but it would be easy to do if needed.

Single Entry Fanout

Another fast way to get to one of many routines is to leave a single entry point, but have BASIC signal which job it wants to do. There are many ways of doing this.

BASIC could POKE a single location with a value that the machine language routines could read and use to take action (favored zero page work locations are hex 0016 to 005D or decimal 22-93).

Another method would be to use the USR argument to tell the machine language routine where to go. USR(1) would go to routine number 1, etc.

Passing Parameters: Single Values

When you call a USR function, the value in parentheses (the argument) gets placed in the floating-point accumulator. When you return from machine language, the value in the floating-point accumulator is accepted by BASIC.

The floating-point accumulator is at locations hex 00A9 to 00AE. It's hard to read if you're not used to floating-point notation.

Location A9 is both exponent and zero flag. If it contains zero, the whole number is zero and you don't need to look any further. If it's non-zero, it's a binary exponent offset by $80. This means that if it contains $80 or less, the number is a fraction less than 1. If it contains $81 or more, the number is greater or equal to 1. Don't worry about the details unless you have a mathematical leaning.. but you double a number by adding one to the exponent, halve it by subtracting one,etc.

The next four locations, AA to AD, contain the number itself, formally called the"mantissa". Because of the way these numbers are arranged, the first bit (high-order bit of AA) should always be 1.

Finally, location AE contains the sign of the number. Only the first bit counts:

if it's zero, the number is positive; if it's one, the number is negative.

Floating-point numbers are nice in BASIC, but they are often overkill in machine language. You will probably want to use the built-in BASIC subroutines to convert to and from the more familiar fixed-point numbers. See the BASIC manual for this.

Multiple Values

This gets tougher. The choice breaks down to this: make it easy on the ML routine and have BASIC poke values to fixed locations; or make it easier on BASIC and have ML dig out the BASIC variables from where they are stored.

Using BASIC to poke and/or peek values is quite straight forward. If your (integer) values go above 255, you'll have to use more than one memory location, of course. Use the standard multiply-or-divide-by-256 technique to separate the bytes or to recombine them.

Going after the BASIC variables directly from machine language is more complex, but often more effective. There are two problems: how to find 'em and then how to read 'em.

Here are a couple of recommendations that may save you some work.

First: Define the variables that you will want ML to pick up early in the program. That way, they will be together and easier to find.

Second: Where possible, use fixed-point BASIC variables. These are the ones with the % sign, such as X% or M3%. Using such variables will save you having to make the conversion from floating-point.

BASIC Variables

Each variable occupies seven locations in memory. Variables start at a location specified by the address in hex 0075-0076 (decimal 117-118), which is stored in the usual manner of low-order first.

You can change the address in 0075-0076 and have the variables start anywhere you choose, but be careful- don't write your variables over the top of something else, such as your BASIC program.

The seven locations for each variable break down as follows: two locations for the variable name, and five for the value.

Let's concentrate on fixed-point variables. The variable name will be in ASCII with the high-order bit set over each of the two name bytes. So B5% will be stored as hex values C2 and B5. C%, which has no second character, will store as C3 and 80. And a long name (not recommended in BASIC) such as CENTRAL% will store only the first two letters as C3 and C5.

The next two locations following the name is the binary value in fixed-point (signed)- high-order first. And the last three locations are not used with fixed-point variables.

So the BASIC statement D9% = 4 will generate the following variable in memory: C4 B9 00 04 followed by the zeros which are not used.

Once you've found a variables location during a run, it won't move; so you don't need to search for it again. Be careful, though: changing your BASIC program will change the locations where the variables will be stored on the next run.

Summary

You can indeed marry BASIC and machine language. This brief article won't give you all the marriage counselling you need; but it may at least perform the introductions.

----------********----------

AIM 65 USER GROUP NORWAY

(Subsidiary to DATAAMATØRFORENINGEN, Oslo)

Meets every second Tuesday, 7PM, at

Konowsgt. 5

Oslo 1

Contact: Per Resell
Dr. Dedichens vei 78
OSLO 6 Telephone (02)269375

# BASIC Bandaid

Dale Hall
19909 Grevillea Ave
Torrance, CA 90503

Let's have lunch while my BASIC program loads.

Problem

Most cassette recorder installations require the interblock gap $A409 be from $40 to $80 to allow time for the token to BASIC statement translation to take place. The printer should not be "ON" during loading as this requires a minimum gap of $80 to avoid loading &/ or printing errors. A list after loading is preferred.

Solution

The token form may be stored using the Monitor command (D), as follows:

1) Enter and create a BASIC program as usual or load in a BASIC program from tape using "LOAD".
2) Exit BASIC...Escape to Monitor.
3) Alter $A409 to $08 or whatever you have been using for object code recording. Some experimentation may be in order. I have used $05 with manual motor control.
4) Use (M) to examine $75,76 AdrL,AdrH for the TO= address.
   Example: $75=ØØ $76=Ø4 TO=Ø4ØØ
5) Dump (D) FROM=75 TO=76 "RET"
   MORE?"Y"
   FROM=212 TO=[76,75] the adr contained in $75,76
   MORE?"N"

The token form may be read into memory using the Monitor (L) command as follows:

1) Initialize BASIC (5) or Re-enter (6) and give the direct command NEW to clear BASIC memory. If you always use the same memory allocation for BASIC, the default value from RET all will work OK. Otherwise some attention to $77 thru $80 is required. Recording $73 thru $80 will cover all the pointers.(Step 5)
2) Proceed as if loading an object file. (L) Load IN=T F=fn. At the conclusion of load the monitor prompt should appear. At very short gaps ($04) it it may not but most of the file will be intact, so hit the hardware reset.
3) Re-enter BASIC (6). That's it.

Note

In auto motor command mode some recorders turn on rather slowly missing the syn characters in the ID block. When this happens the SRCH and LOAD never are obtained. Try disabling the auto controls and start the motor manually, before hitting RET when recording and vice-versa in playback. Much shorter interrecord gaps may be achieved in manual operation.

Text Editor program's may similiarly be stored at $E1 thru $E6. Since the beginning and ending addresses may be anywhere in available memory, they must be examined to place them in the prompted input as well as recorded as was done in BASIC. However this will load more slowly than the "L" command with an identical $A4Ø9 count due to formatting overhead in the dump routine. Mentioned for academic curiousity types!

Sample follows:

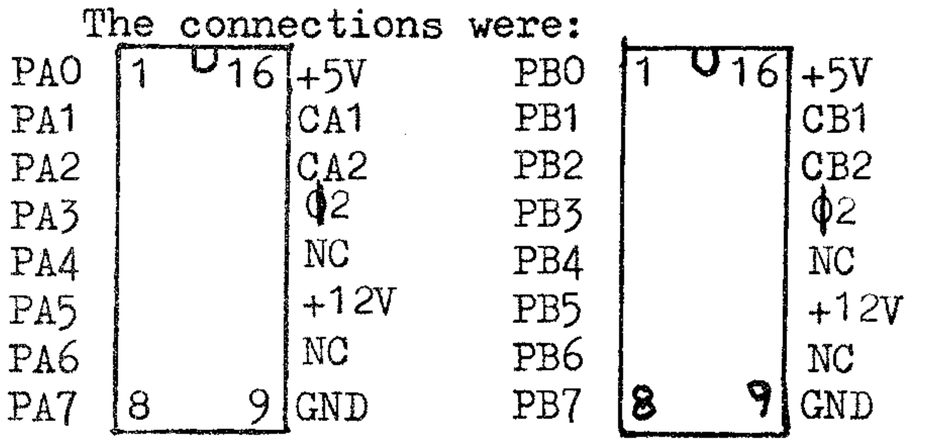
```
(M)=073 12 02 40 07
( ) 0077 40 07 40 07
( ) 007B 00 50 76 4C
( ) 007F 00 50 28 FF
( ) 0083 28 00 3F 00

(D)
FROM=73  TO=80
OUT=T F=C%$20 T=1
MORE?Y
FROM=0212  TO=0740
MORE?N
(M)=A409  20 04 CA 0
6
(/) A409 08
(D)
FROM=73  TO=80
OUT=T F=C%$08 T=1
MORE?Y
FROM=0212  TO=0740
MORE?N
```

# User VIA

Steve Bresson
1302 Strawberry Lane
Hanover, MD 21076

I ran a cable from the application connector to 2 16 pin sockets that are mounted on the back of my case.

The connections were:




This makes it easy to connect things to the USER VIA- you merely plug in a 16 pin dip plug. This also makes it simple to change the use of each I/O port.

For Lunar Lander you plug in a socket with pins 7 and 8 shorted together, and a for SIMON game, you put a plug with the lower four bits as inputs, and the upper four bits as outputs.

CHANGES TO THE AIM

Here are some reader supplied changes to the AIM 65 that they would like to see.

More conservative use of address space. An extra VIA too. 1 x 20 display is fine. User expansion of BASIC commands. A thermal printer whose output lasts. 40 character display. Wished BASIC talked to cassette easier. Longer line lengths. On board video.

PERIPHERALS

The most common accessory attached to Target readers was memory. Teletypes and video displays were also being used. Two peripherals which deserve special attention because they are quite unusual. They are a 6MeV Van de Graaf accelerator and a cosmic-ray monitoring station. Now that is what I call unusual! The same person has both of these. Can you top these?

# Memory

Steve Roberts
7100 Mimosa Drive
Carlsbad, CA 92008

I wanted to expand my AIM 65 to 12K using the HDE DM816-M8 memory board. This board is of excellent quality and has further advantages of reasonable price, small size, and KIM bus compatibility. Unfortunately, as delivered from the factory it can be assigned to memory space in 8K blocks starting on even address boundaries only ( the 4K version of the board can start on odd or even boundaries). Since I had 4K on board the AIM, I wanted to start the new 8K block of memory at address $1000.

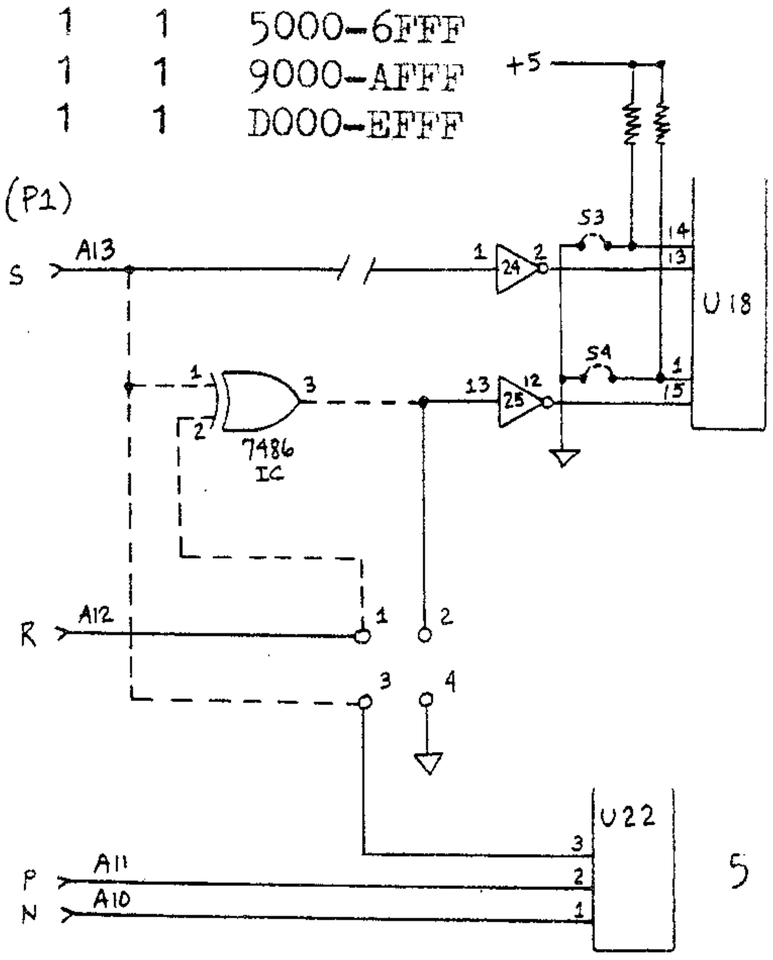
The modification shown allows the 8K board to start at $1000, $5000, $9000, or $D000. This solved my particular problem, but hopefully someone will improve the modification to be able to start at any odd or even boundary. This memory modification has been in essentially continuous use for approximately 4 months with no problems.

Modifications

Dashed lines are new connections. The board trace for A13 is cut, and pin 1 on IC24 is left floating. No jumpers are placed at points 1, 2, 3, or 4. The added IC is a 7486 EXCLUSIVE OR. The last two switch settings should not be used, due to conflicts with system addresses.

--Switch Settings--

| Switches (1) | (2) | (3) | (4) | Memory Decoded |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1000-2FFF |
| 0 | 1 | 1 | 1 | 5000-6FFF |
| 1 | 0 | 1 | 1 | 9000-AFFF |
| 1 | 1 | 1 | 1 | D000-EFFF |



# Letters

Jody Nelis
132 Autumn Drive
Trafford, PA 15085

In the 2-1/2 years I've had a KIM and an AIM, I,ve used machine language pretty much exclusively. Various sources such as 6502 User Notes and Micro have unlocked a lot of the mystery in the respective monitors of these machines.

Subscribing to Target and seeing that it is nearly 100% BASIC oriented inspired me to purchase ROM BASIC for the AIM. I had great hopes that Target would similiarly unlock the mystery of running BASIC. So far it's not happening that way. Most Target articles seem to assume the reader knows a great deal already. Too many of them shoot over my head. If a little more were included or a few instructions included, I think I would grasp more- I come close but no cigar on many of the ideas presented.

As an example or two- take the May/June 1980 issue, The idea on page 1 from George Sellers looks good but how does one know the address of the second BASIC program statement to allow fixing $0212 and 0213?? Maybe some digging into other sources would tell me this, but maybe a few more thoughts with this comment.would do the same.

Steve Bresson's Renumber in the same issue is loaded with example runs and lists, but other than the left column on page two, the rest of the examples mean nothing to me. What is happening with the listings showing "NO ROOM TO REPLACE XX WITH XXX"?? Why not? What to do now? How does one load RENUM in RAM with the application program other than type it in by hand each time? The delete routine shows how to get rid of it (I think) but now how to load two programs into BASIC from tape. Again- maybe more digging would answer but maybe a reference or a few words here would do it also.

In general, I would like to see the AIM utilities and user hints spelled out a little more thoroughly to make them suitable for use by someone like me. The Rockwell BASIC manual sure doesn't give any hints!

Remember that someone with experience can rapidly read through the simple stuff to get to the meat of an article but someone like me without experience can't fill in the missing lines!!

Renumber Comments

Steve Silber
5815 Southminster
Houston, TX 77035

I was very pleased to see Steve Bresson's BASIC renumbering program (May/June 1980). I have two observations to make in regard to his program, and would also mention in passing that I am working (slowing, it seems) on a machine language tape to tape version that hopefully will be finished by the end of summer.

The first point in reference to Bresson's program is that Microsoft BASIC recognizes GO TO as being legal and distinct from GOTO. Bresson's program does not allow for this, but a couple extra lines should solve the problem. GO TO is stored as three (or more) bytes: $C5,$20, ($20,$20...),$E9. One possible patch might be:

```
63539 IF CH=197 THEN 63542  (test for GO code)
63541 GOTO 63550            (normal cont)
63542 L=L+1:CH=PEEK(L):IF CH=32 THEN 63542
                            (skip imbedded spaces)
63543 IF CH<>233 THEN 63530 (go back if
                             not TO)
```

The second thing I would like to point out in connection with the Bresson article is that it is unnecessary to go through all the trouble he did in order to save only the renumbered program and not the renumbering program above 63000. Although not mentioned in the BASIC manual, the BASIC interpreter will execute the command SAVE-62999, which is exactly what he needed. In fact, the interpreter uses the same code to do a SAVE as it does to do a LIST, and the only reason that it will not accept a command such as SAVE100-1000 (?SN ERROR) is that the carry flag is not preserved when the tape output file is opened.

(ED. Note- Steve Bresson comments that he did not know the two items mentioned above so they were not included in his article or program. Steve also suggests using the following form to test for GO TO instead of the one listed from Steve S.

Replace lines 63539 and 63541 with:
63539 IF CH<>197 THEN 63550)

# Tips'etc.

Michael Rathbun
PO Box 268
Kodiak, AK 99615

Here is a quick-and-dirty slow display for AIM systems not using a teletype.

Jumper Application connector pin R to T.
Change location A417 to 80(or convenient value).

When slow display is desired, key in all but the last keystroke of the desired command.
Set KB/TTY switch to TTY.
Key in last keystroke of desired command.
Slow display will commence.

Disadvantage- When using this for BASIC LIST, AIM keyboard does not regain control unless KB/TTY switch is returned to KB before last character is displayed. If system hangs up in this manner, it is necessary to use reset, or to momentarily interrupt the jumper ( normally-closed switch handy for this) after setting switch to KB.

Advantage- Consumes no memory or I/O resource; requires no program to be loaded, since the timing loop is part of the ROM monitor.

Christopher Flynn
2601 Claxton Drive
Herndon, VA 22070

Shortly after I got my AIM (from Excert) the printer started making grinding noises and printing terribly. When I was able to look at the situation calmly, I tried putting a little light grease on the lateral contact areas. That cured the problem. I won't tell you about the time I nearly.....

*6502 Assembly Language Programming by Lance Lenthal from Osborne/Mc-Graw, Hill $12.50 plus $1.00 handling (non-US orCAN $4.00) C/O Donald Clem

Best of 1979 will be along shortly!!!

Rich Ledford
1217 S. Wenonah
Berwyn, Ill 60402

I am interested if anyone has "Tiny C" running on an AIM 65 with memory expansion and/or disk.

Ron Riley
PO Box 4310
Flint, MI 48504

Readers of Target may be interested in the fact that Sears Model 272-58020 printing calculator uses a printer that is the same as AIM 65 except for the mounting feet.

This means that AIM users should be able to buy printer parts from Sears and thermal paper #3-3986 can be used on the AIM. (Ed. Note- Rockwell recomends Sears paper #3-3974)

## GENERAL INFORMATION

Article contributions are always welcome. Program listings may or may not be retyped. When submitting information on AIM thermal paper adjust the darkness control to its darkest setting. Artwork will not be redrawn so please submit your best work. Artwork may be oversize if necessary and will be reduced to proper size.

Text should accompany articles to explain what is being done, how it is done, and how it may be modified to suit the user.

Please submit a self addressed stamped envelope for any replies that you desire.

Back Issues- A consolidated 1979 issue is available for $6.00 ($12). In addition 1980 issues are available beginning with the January/ February and at subsequent two month **intervals**. Individual 1980 issues are $1.00 (US and CAN, $2.00 elsewhere).

Time to Renew- The mailing label contains the last issue that you will receive. If no date appears you have at least two issues left on you subscription.

Target- an AIM 65 newsletter is published bimonthly with an annual subscription rate of $6.00 in the US and Canada, $12.00 elsewhere. Contact Donald Clem, RR# 2, Spencerville,OH 45887.

# Directory

Steve Silber
5815 Southminster
Houston, TX 77035

This program is designed to inventory the contents of AIM 65 format tapes. The program asks for the filename of the last file to be listed, and the tape number. It will then proceed to read the tape and list all the files found, up to and including the one specified. If no filename(or non-existant filename) is given the program will continue searching indefinitely, until the reset button is pressed.

It may be a good idea to save a DUMMY file at the end of each tape in order to have a file to search to. As each file is found, the program will print its filename and type, whether OBJECT, SOURCE, or BASIC. If the file is a text file (source or BASIC), the program will count the number of lines of text and print that at the end of the file. If the file is an object file, the program will read the address blocks and print out the contiguous RAM areas into which the program would normally load. It also verifies the block checksums. After every file, the program prints out the number of tape blocks the file contains. Note that this number is one greater than the highest block number since the first block is $00.

Comments

This program can be very useful, both for keeping a record of tape contents, and for sorting out old scratch tapes with unknown data on them. If your recorder has a tape counter on it, note the reading at the beginning of each file, and make it easy to relocate a file in the future.

It is especially nice to know in advance what areas of RAM will be affected by an object program file load, especially if you have two or more versions of a program that load into different areas of RAM. The tape directory folds up nicely into the plastic cassette storage cases, making it very easy to associate a tape and directory together.

Note that KIM format tapes will not read or list.

The Program

$200-$216 inputs the filename and tape number. LOOP1 at $217 looks for tape block number 00, indicating a new file. LOOP2 at $226 prints a five character filename from the tape buffer, and LOOP3 at $239 compares the name to the input name and sets the end flag if they match. Next the character after the filename is examined and if equal to $0D then an object file is indicated. Object files are read much the same way that the tape verify routine does, except that a record is kept of the destination memory address.

If the file is a text file, it's type (source or BASIC) is defined by whether or not the carriage returns are followed by linefeeds, since BASIC outputs a line-feed character and the text editor does not. In either case, the file type is printed and the number of lines determined by counting either carriage returns or carriage return-linefeed combinations. In either case the end of file is indicated by a null line(double CR or CR-LF). The number of lines is printed, followed by the number of blocks. If the end flag is not set, the program returns to the top at LOOP1 to search for another file.

I am willing to make copies of this program or Auto-Number Short Cut(from the March/April 1980 Target) if a reader supplies a blank cassette and $2.00. If desired, this would include re-assembly to run at any specified address. Be sure to specify what tape gap you normally use for source files.

```
ASSEMBLER


LIST?
LIST-
PASS 1

PASS 2

TAPE DIRECTORY PROGRAM
BY STEVE SILBER
MONITOR EQUATES

==0000 RDBLK=$ED53
==0000 OUTPUT=$E97A
==0000 CLRCK=$EB4D
==0000 CHEKAR=$E54B
==0000 RBYTE=$E3FD
==0000 BLANK2=$E83B
==0000 NUMA=$EA46
==0000 CKSUM=$A41E
==0000 INALL=$E993
==0000 WRAX=$EA42
==0000 CRLF=$E9F0
==0000 FNAM=$E8A2
==0000 ADDR=$A41C
==0000 TAPTR=$A436
==0000 TBUFF=$0116
==0000 BLK=$0115
==0000 FNAME=$A42D
==0000 INFLG=$A412

--------------------
PAGE ZERO REGISTERS

==0000
        *=$00F0
==00F0 ENDFLG
        *=*+1
==00F1 TEMP1
        *=*+2
==00F3 TEMP2
        *=*+2
==00F5 TEMP
        *=*+1
==00F6 LNCNT
        *=*+2

--------------------
F1 LINKAGE

==00F8
        *=$10C
==010C
4C0002 JMP INIT

--------------------
MAIN PROGRAM

==010F
        *=$200
```

```
PAGE 02

==0200 INIT
A954    LDA #'T
8D12A4 STA INFLG
;SET INPUT TO TAPE
A200    LDX #0
20B503 JSR MPRNT
;ASK FOR FILE & TAPE
A200    LDX #0
86F0    STX ENDFLG
20A2E8 JSR FNAM
==0211
20F0E9 JSR CRLF
20F0E9 JSR CRLF
==0217 LOOP1
A200    LDX #0
8E1501 STX BLK
2053ED JSR RDBLK
CA      DEX
BD1601 LDA TBUFF,X
D0F2    BNE LOOP1
;WAIT FOR BLOCK 0
E8      INX
==0226 LOOP2
BD1601 LDA TBUFF,X
207AE9 JSR OUTPUT
E8      INX
E006    CPX #06
D0F5    BNE LOOP2
;PRINT FILE NAME
8E36A4 STX TAPTR
;SAVE POINTER
203BE8 JSR BLANK2
==0237
A201    LDX #1
==0239 LOOP3
BD1601 LDA TBUFF,X
DD2DA4 CMP FNAME,X
D009    BNE OKAY
E8      INX
E006    CPX #6
D0F3    BNE LOOP3
;TEST IF LAST FILE
A9FF    LDA #$FF
85F0    STA ENDFLG
==024A OKAY
AE36A4 LDX TAPTR
BD1601 LDA TBUFF,X
C90D    CMP #$0D
;TEST IF OBJECT FILE
F003    BEQ *+5
4C3403 JMP TEXT
A212    LDX #MES2-MES1
20B503 JSR MPRNT
==025C
20F0E9 JSR CRLF
==025F LOOP4
2093E9 JSR INALL
C93B    CMP #':
D0F9    BNE LOOP4


PAGE 03

204DEB JSR CLRCK
204BE5 JSR CHEKAR
AA      TAX
204BE5 JSR CHEKAR
==0270
8D1DA4 STA ADDR+1
204BE5 JSR CHEKAR
8D1CA4 STA ADDR
;GET COUNT & ADDRESS
86F1    STX TEMP1
AA      TAX
BD1DA4 LDA ADDR+1
2042EA JSR WRAX
==0282
A248    LDX #MES9-MES1
20B503 JSR MPRNT
A6F1    LDX TEMP1
==0289 LOOP5
204BE5 JSR CHEKAR
CA      DEX
F008    BEQ OVER
EE1CA4 INC ADDR
D003    BNE *+5
EE1DA4 INC ADDR+1
;COUNT THROUGH BLOCK
;& INCREMENT ADDRESS
==0297 OVER
8A      TXA
D0EF    BNE LOOP5
20FDE3 JSR RBYTE
CD1FA4 CMP CKSUM+1
D065    BNE ERR
20FDE3 JSR RBYTE
CD1EA4 CMP CKSUM
==02A8
D05D    BNE ERR
;CHECK CHECKSUMS
2093E9 JSR INALL
C93B    CMP #':
D0F9    BNE *-5
204DEB JSR CLRCK
204BE5 JSR CHEKAR
AA      TAX
==02B8
204BE5 JSR CHEKAR
85F2    STA TEMP1+1
204BE5 JSR CHEKAR
85F1    STA TEMP1
8A      TXA
F04A    BEQ END
38      SEC
85F5    STA TEMP
==02C8
;TEST IF CONTIGUOUS
;MEMORY BLOCKS
A5F1    LDA TEMP1
ED1CA4 SBC ADDR
85F3    STA TEMP2
A5F2    LDA TEMP1+1


PAGE 04

ED1DA4 SBC ADDR+1
85F4    STA TEMP2+1
D008    BNE MOVE
==02D8
9006    BCC MOVE
A5F3    LDA TEMP2
C901    CMP #01
F018    BEQ OK2
==02E0 MOVE
AD1DA4 LDA ADDR+1
AE1CA4 LDX ADDR
2042EA JSR WRAX
20F0E9 JSR CRLF
A5F2    LDA TEMP1+1
A6F1    LDX TEMP1
==02F0
2042EA JSR WRAX
A248    LDX #MES9-MES1
20B503 JSR MPRNT
;CLOSE LAST & OPEN
;NEW LISTING BLOCK
==02F8 OK2
A5F1    LDA TEMP1
8D1CA4 STA ADDR
A5F2    LDA TEMP1+1
8D1DA4 STA ADDR+1
A6F5    LDX TEMP
4C8902 JMP LOOP5
==0307 ERR
A223    LDX #MES6-MES1
20B503 JSR MPRNT
4C1702 JMP LOOP1
==030F END
AD1DA4 LDA ADDR+1
AE1CA4 LDX ADDR
2042EA JSR WRAX
==0318 END2
20F0E9 JSR CRLF
AD1501 LDA BLK
2046EA JSR NUMA
A241    LDX #MES5-MES1
20B503 JSR MPRNT
20F0E9 JSR CRLF
==0329
20F0E9 JSR CRLF
A5F0    LDA ENDFLG
3003    BMI *+5
4C1102 JMP LOOP1-6
60      RTS
==0334 TEXT
A900    LDA #00
85F6    STA LNCNT
85F7    STA LNCNT+1
;INITIALIZE LINE CNT
==033A TLOOP
2093E9 JSR INALL
C90D    CMP #$0D
D0F9    BNE TLOOP
;WAIT FOR <CR>


PAGE 05

20A003 JSR INCR
2093E9 JSR INALL
C90A    CMP #$0A
;<LF> IS BASIC
F033    BEQ BASIC
==034B
48      PHA
A218    LDX #MES3-MES1
20B503 JSR MPRNT
20F0E9 JSR CRLF
68      PLA
C90D    CMP #$0D
F011    BEQ EXIT
==0359 LOOP6
2093E9 JSR INALL
C90D    CMP #$0D
D0F9    BNE LOOP6
;COUNT <CR>S
20A003 JSR INCR
2093E9 JSR INALL
C90D    CMP #$0D
D0EF    BNE LOOP6
==036A
;LOOK FOR DOUBLE
==036A EXIT
A5F7    LDA LNCNT+1
F003    BEQ *+5
2046EA JSR NUMA
A5F6    LDA LNCNT
2046EA JSR NUMA
A231    LDX #MES7-MES1
20B503 JSR MPRNT
==037B
;PRINT SUMMARY&EXIT
4C1803 JMP END2
==037E BASIC
A21E    LDX #MES4-MES1
20B503 JSR MPRNT
20F0E9 JSR CRLF
==0386 LOOP7
2093E9 JSR INALL
C90D    CMP #$0D
D0F9    BNE LOOP7
20A003 JSR INCR
2093E9 JSR INALL
C90A    CMP #$0A
;BASIC COUNTS <CR>
;<LF> COMBINATIONS
D019    BNE TERR
==0397
2093E9 JSR INALL
C90D    CMP #$0D
D0E8    BNE LOOP7
;DOUBLE FOR END
F0CA    BEQ EXIT
;
;SUBROUTINE FOR THE
;DECIMAL LINE COUNT
;


PAGE 06

==03A0 INCR
F8      SED
18      CLC
A5F6    LDA LNCNT
6901    ADC #1
85F6    STA LNCNT
A5F7    LDA LNCNT+1
6900    ADC #0
85F7    STA LNCNT+1
D8      CLD
60      RTS
==03B0 TERR
A237    LDX #MES8-MES
4C0903 JMP ERR+2
;
;MESSAGE PRINTER
;

==03B5 MPRNT
BDC303 LDA MES1,X
3006    BMI OUT
207AE9 JSR OUTPUT
E8      INX
D0F5    BNE MPRNT
==03C0 OUT
4C7AE9 JMP OUTPUT
==03C3 MES1
4C41    .BYTE 'LAST FILE' & TAPE ',$A3
==03D4
A3
==03D5 MES2
4F42    .BYTE 'OBJEC',$D4
D4
==03DB MES3
534F    .BYTE 'SOURC',$C5
C5
==03E1 MES4
4241    .BYTE 'BASI',$C3
C3
==03E6 MES6
4348    .BYTE 'CHECKSUM ERRO',$D2
D2
==03F4 MES7
204C    .BYTE ' LINE',$D3
D3
==03FA MES8
5445    .BYTE 'TEXT ERRO',$D2
D2
==0404 MES5
2042    .BYTE ' BLOCK',$D3
D3
==040B MES9
2D2D    .BYTE '--->',$A0
A0
        .END
 ERRORS= 0000
 LABELS= 0055
ADDR    =A41C
BASIC   =037E
BLANK2  =E83B


  PAGE 07

  BLK       =0115
  CHEKAR    =E546
  CKSUM     =A41E
  CLRCK     =EB4D
  CRLF      =E9F0
  END       =030F
  END2      =0318
  ENDFLG    =00F0
  ERR       =0307
  EXIT      =036A
  FNAM      =E8A2
  FNAME     =A42D
  INALL     =E993
  INCR      =03A0
  INFLG     =A412
  INIT      =0200
  LNCNT     =00F6
  LOOP1     =0217
  LOOP2     =0226
  LOOP3     =0239
  LOOP4     =025F
  LOOP5     =0289
  LOOP6     =0359
  LOOP7     =0386
  MES1      =03C3
  MES2      =03D5
  MES3      =03DB
  MES4      =03E1
  MES5      =0404
  MES6      =03E6
  MES7      =03F4
  MES8      =03FA
  MES9      =040B
  MOVE      =02E0
  MPRNT     =03B5
  NUMA      =EA46
  OK2       =02F8
  OKAY      =024A
  OUT       =03C0
  OUTPUT    =E97A
  OVER      =0297
  RBYTE     =E3FD
  RDBLK     =ED53
  TAPTR     =A436
  TBUFF     =0116
  TEMP      =00F5
  TEMP1     =00F1
  TEMP2     =00F3
  TERR      =0380
  TEXT      =0334
  TLOOP     =033A
  WRAX      =EA42
```

# New Product

Macro-assemblers give the user the ability to enhance or re-define the set of mnemonics which the assembler recognizes, with the added advantage that a few or a hundred lines of pre-defined source statements can be inserted into the source program being assembled with just a few keystrokes. With a macro-assembler, you can simplify complex, repetetive programming tasks, cross-assemble programs for some other type of processor (like the RCA 1802), and even develope your own compiler-like program generation system.

MACRO is a RAM resident program which takes advantage of the versatile AIM editor and cassette interfaces to emulate macro-assembler operation by pre-processing source files to generate intermediate files which can then be processed by the AIM assembler.

With a suitable library of macro sequences, you may find it possible to generate a 200-line assembly-level program by keying in only a few dozen lines.

MACRO costs $15.00 for an object cassette ($30.00 for source) and includes a user's guide with programming examples. The user's guide alone may be purchased for $2.50. Contact POLAR SOLUTIONS, Box 268, Kodiak, Alaska 99615.





THE TARGET
c/o DONALD CLEM
R.R. #2, CONANT RD.
SPENCERVILLE, OHIO 45887